Technical Disclosure Commons

Defensive Publications Series

December 11, 2017

Handwriting recognition for IDEs with Unicode support

Michal Luszczyk

Sandro Feuz

Handwriting recognition for IDEs with Unicode support

ABSTRACT

This disclosure describes techniques for handwriting recognition for Unicode text entered via touch or other input. For example, Unicode text may be part of program source code in some programming languages. The handwritten code is provided as an input to a trained machine learning model. The model produces possible matches for the user entered text along with a score for each match. The user, e.g., a programmer, can select the appropriate match which is then added to the source code. The model is trained with sample handwritten text for known programs, and can also utilize additional factors such as program context and handwriting stroke information. The techniques can be implemented in IDE software, operating systems, and in devices that enable such input, e.g., electronic whiteboards.

KEYWORDS

- Handwriting recognition
- IDE
- Integrated development environment
- Input method
- Unicode
- Programming environment
- Electronic whiteboard

BACKGROUND

Computer programmers commonly utilize Integrated Development Environments (IDE) to write and review software code. Current IDEs are mostly designed for traditional desktop computers and laptops. With increased use of devices such as tablets, electronic whiteboards, etc.

that include capabilities for handwriting, stylus, and gesture-based inputs, enabling programming environments on these devices can enhance user experience. Support for entering Unicode symbols in programming and development environments is useful for users of programming languages that support Unicode symbols, such as Haskell, Python, and Java.

DESCRIPTION

This disclosure describes an Integrated Development Environment (IDE) that incorporates handwriting recognition for entering program source code. A machine learning model is deployed for the handwriting recognition. The model is trained to interpret Unicode characters, e.g., that are used in source code.

Fig. 1 illustrates the use of the handwriting recognition within a programming environment or IDE (110). A text box (120) is provided, where a programmer can input code using touch or stylus input, e.g., using a touchscreen or other touch-sensitive surface. In the example shown in Fig. 1, the programmer has provided four lines of handwritten text (122-128) for a Haskell program. Once the user provides the input, a trained machine learning model is used to evaluate the partial or complete line of program code.

The recognition model returns interpreted text (130) that includes multiple interpreted matches. In the example shown in Fig.1, two matches (132, 134) for the second line of text (124) are shown, along with respective probability scores that indicate a confidence level for each match. The first match (132) is associated with a score of 95%. The second match (134) includes an alternate interpretation of the user entered program code and is associated with a lower score of 25%. Fig. 1 shows matches for a single line as an example, the techniques are also applicable to blocks of code that include multiple lines.

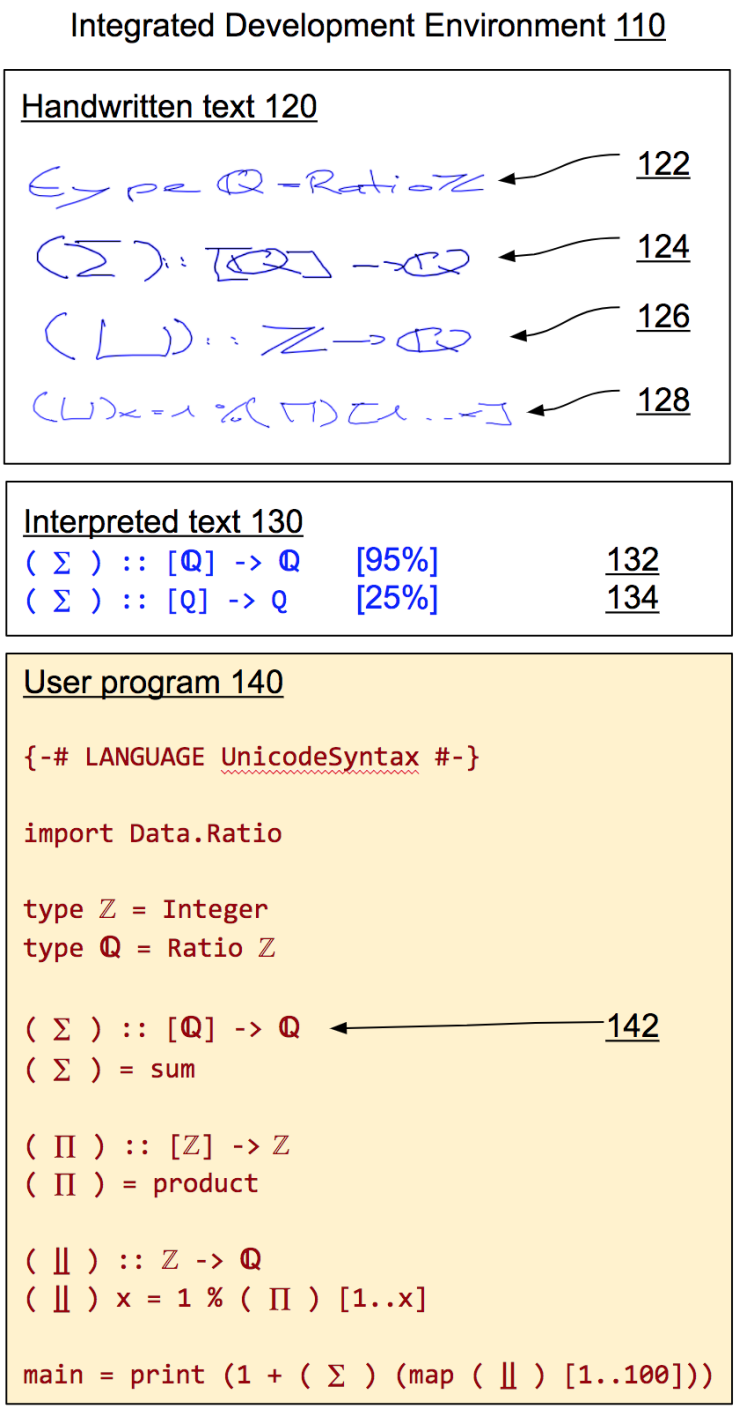

Fig. 1: Handwriting recognition and probability scores

The user can then select the correct match from the candidate matches, which is then added (142) to the user program (140) in the IDE. In a collaborative environment, multiple users

can enter program code via touch simultaneously. Each user entry is interpreted separately with the handwriting recognition model.

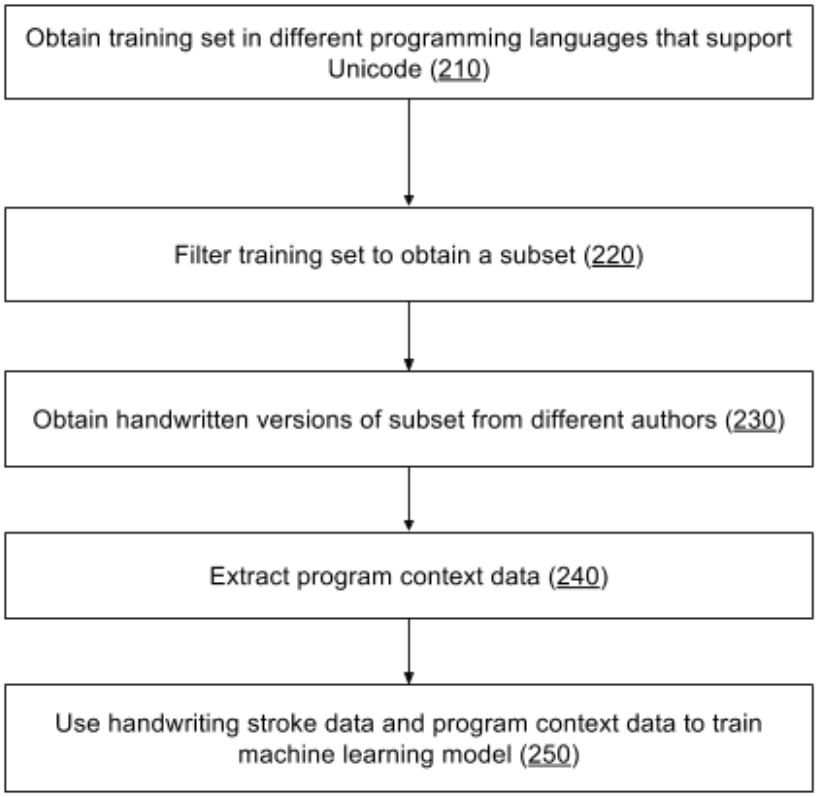

Fig. 2: Training a machine learning model for Unicode handwriting recognition

Fig. 2 illustrates an example training process for the machine learning model for the handwritten source code. A training set is obtained (210) by using a set of programs in different programming languages supporting Unicode characters, e.g., Haskell, Python, Java, etc. This set of programs is filtered (220) to create a subset of programs that each include at least a certain threshold of Unicode characters.

Handwritten samples of these programs from different authors are obtained, with user permission (230). Additional signals such as local and global variables, syntax rules of the respective programming language, declared functions, and methods of the respective language,

etc., are inferred (240) from the context of the program. A machine learning model, e.g., a feature-based model or a recurrent neural network such as long short-term memory (LSTM) or gated recurrent units (GRU) model, etc., can be trained (250) using the training data. Stroke information of the handwritten source code, syntactical information about the programming language (i.e. language model) and the additionally collected signals from the program context are provided as inputs to the machine learning model.

While the foregoing discussion describes a trained machine learning model, heuristics can also be used to recognize the Unicode text. The techniques described herein can be incorporated into IDE software, and can also be offered as part of operating systems, e.g., mobile operating systems. For example, the techniques can be implemented via a system level application programming interface (API) that application developers can access. Electronic whiteboards that enable collaboration and include touch capability, can also include Unicode source code recognition.

CONCLUSION

This disclosure describes techniques for handwriting recognition for Unicode text entered via touch or other input. For example, Unicode text may be part of program source code in some programming languages. The handwritten code is provided as an input to a trained machine learning model. The model produces possible matches for the user entered text along with a score for each match. The user, e.g., a programmer, can select the appropriate match which is then added to the source code. The model is trained with sample handwritten text for known programs, and can also utilize additional factors such as program context and handwriting stroke information. The techniques can be implemented in IDE software, operating systems, and in devices that enable such input, e.g., electronic whiteboards.